# ARQUITECTURA Y DISEÑO DE SISTEMAS

# ATRIBUTOS DE CALIDAD – PARTE 1

**ELSA ESTEVEZ**
UNIVERSIDAD NACIONAL DEL SUR
DEPARTAMENTO DE CIENCIAS E INGENIERIA DE LA COMPUTACION

# CONTENIDO

# TACTICAS

MOTIVACIÓN

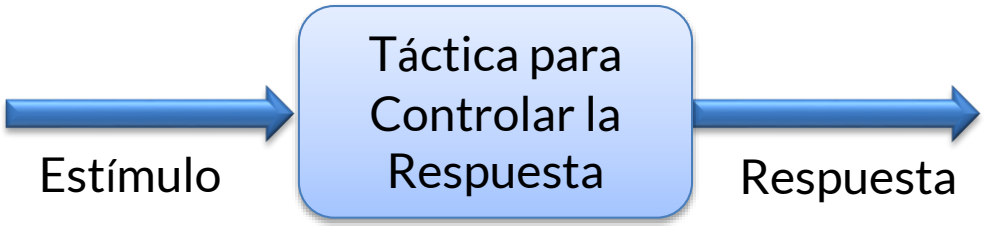
¿Cómo se imparte portabilidad a un diseño?
¿Cómo se imparte performance a un diseño?
¿Cómo se imparte interoperabilidad a un diseño?
…

## DEFINICIÓN

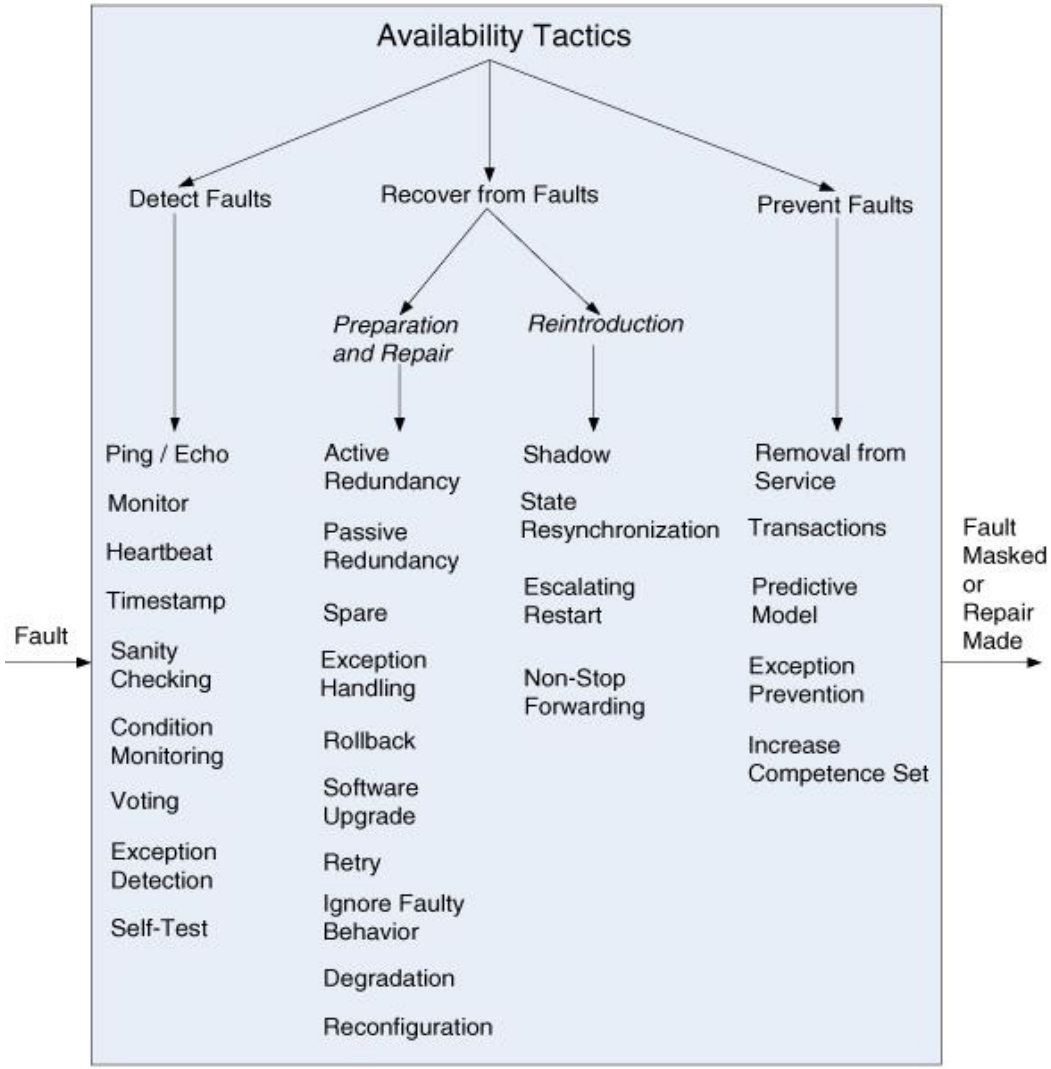
Una táctica es una decisión de diseño que influencia el control de la respuesta de un atributo de calidad

Consideraciones:

- Cada táctica es una opción de diseño. Puede haber otras.
- Una táctica puede ser refinada en otras
- Generalmente las tácticas para un atributo de calidad se organizan en una jerarquía



# TÁCTICAS - EJEMPLOS



# MODIFICABILIDAD

## DEFINICIÓN

Es el costo (riesgo) asociado a realizar un cambio en un sistema.

## ÁREAS DE INTERÉS

| | | |
|---|---|---|
| ¿Qué puede cambiar? | ○ Funcionalidad<br>○ Plataforma – hardware, sistema operativo, middleware<br>○ Ambiente – sistemas con los que interopera<br>○ Calidades del sistema (performance)<br>○ Capacidad (número de usuarios, número de transacciones) | |
| ¿Cuándo se hace el cambio? | Codificación<br>Compilación<br>Construir paquete (Build) | Configuración<br>Ejecución |
| ¿Quién hace el cambio? | Desarrollador<br>Administrador del sistema<br>Usuario final | |
| ¿Cuál es la probabilidad de cambio? | | |

# MODIFICABILIDAD – ANÁLISIS

Cuál es el costo de hacer un sistema más modificable?

- ✓ El costo de introducir el/los mecanismo/s para hacer el sistema más modificable
- ✓ El costo de hacer la modificación usando el/los mecanismo/s

| | Sin mecanismo | Con mecanismo |
|---|---|---|
| Costo de introducir el mecanismo | Cero | Costo de crear el mecanismo |
| Costo de hacer la modificación | El costo de cambiar el código fuente y testearlo | Costo de usar el mecanismo para hacer el cambio y testear el cambio |

Para N modificaciones similares, una justificación simplificada para la creación de un mecanismo de cambios podría ser:

N * costo de hacer el cambio sin el mecanismo => costo de crear el mecanismo + (N * costo de hacer el cambio usando el mecanismo)

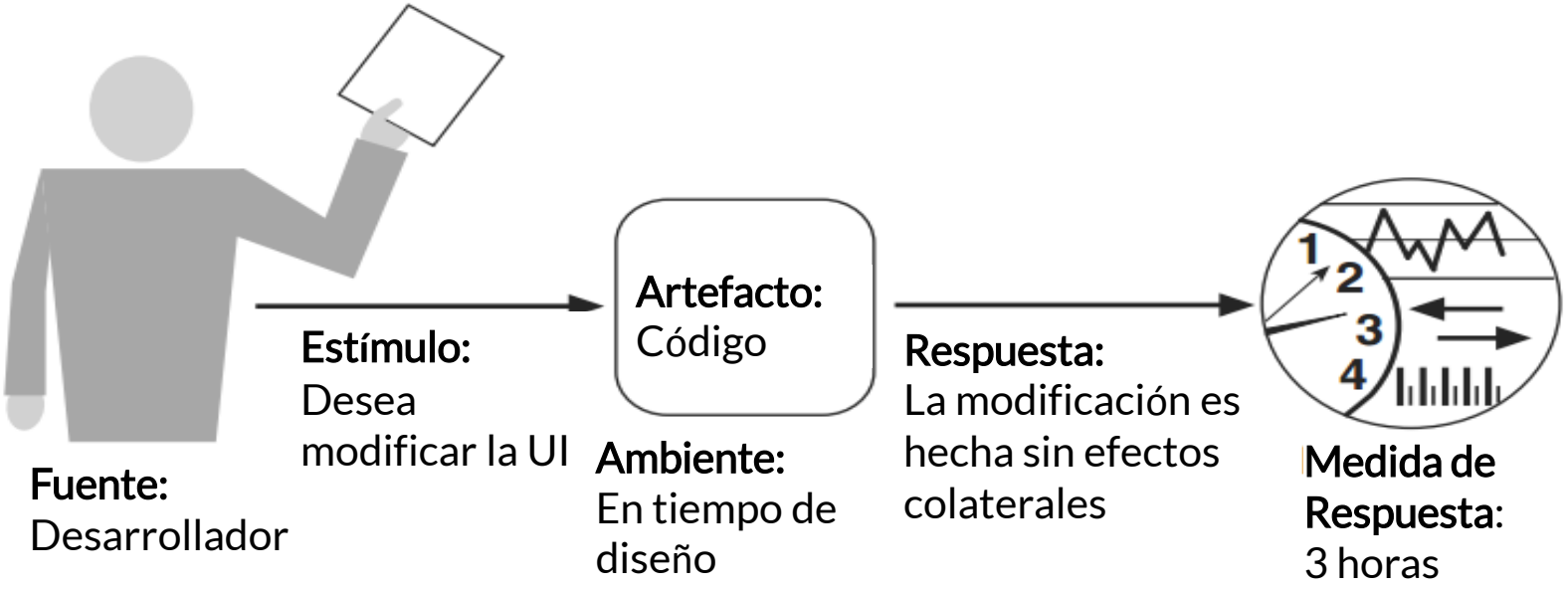
N es la cantidad de modificaciones del mismo tipo que se prevén (una predicción)

# MODIFICABILIDAD – EJEMPLO DE ESCENARIO GENERAL

| PARTE | VALORES POSIBLES |
|---|---|
| FUENTE | [Quién hace el cambio]<br>Usuario final, desarrollador, administrador del sistema |
| ESTÍMULO | [Qué cambio debe hacerse]<br>Deseo de agregar/modificar/eliminar funcionalidad, atributos de calidad… |
| ARTEFACTO | [Qué tiene que cambiar]<br>UI del sistema, plataforma, ambiente, sistemas con los que interactúa |
| AMBIENTE | [Cuándo se hace el cambio]<br>Ejecución, compilación, build, diseño |
| RESPUESTA | ▪ Localizar los lugares a ser modificados<br>▪ Realizar las modificaciones sin afectar otras características del sistema<br>▪ Testear las modificaciones<br>▪ Instalar modificaciones |
| MEDIDA DE LA RESPUESTA | Costo del cambio en términos de:<br>▪ Elementos afectados<br>▪ Esfuerzo<br>▪ Dinero<br>▪ Grado en que afecta a otras características del sistema |

# MODIFICABILIDAD – EJEMPLO DE ESCENARIO CONCRETO

"Un desarrollador desea cambiar la interfaz de usuario para que el color de fondo de la pantalla sea azul. El cambio se realizará sobre el código en tiempo de diseño, debiendo tomar no más de 3 horas en realizarlo y testearlo y sin tener efectos colaterales sobre el comportamiento del sistema."

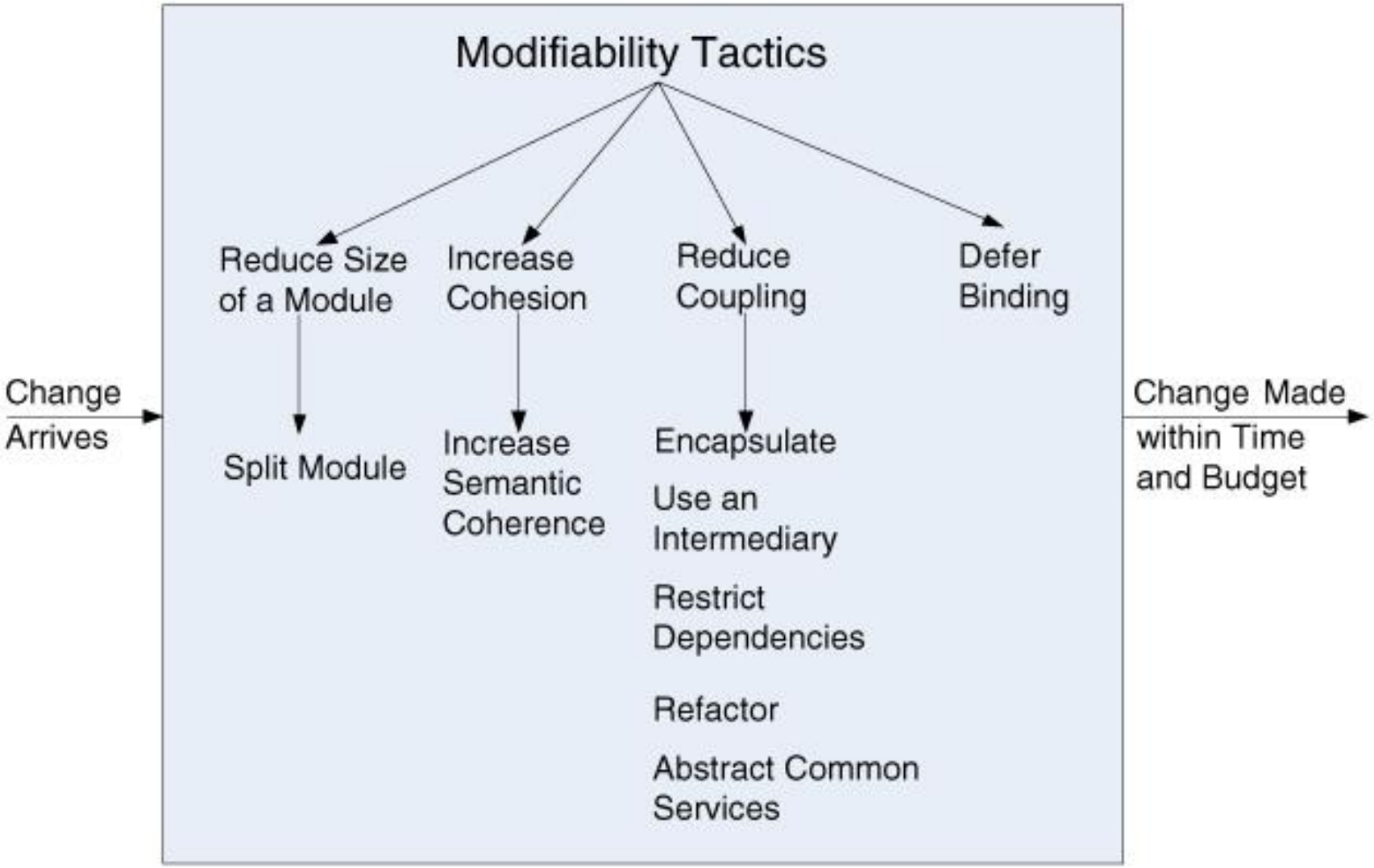


# MODIFICABILIDAD – CONSIDERACIONES PARA TÁCTICAS

Se basan en 4 variables:

- TAMAÑO DEL MÓDULO
  Siempre y cuando el criterio para dividir el módulo refleje el tipo de cambio que es probable de ocurrir
- ACOPLAMIENTO
  Reducir el acoplamiento entre dos módulos A y B disminuirá el costo esperado de cualquier modificación que afecte a uno de esos módulos.
- COHESIÓN
  Si un módulo tiene una baja cohesión, puede ser mejorada eliminando responsabilidad no afectadas por los cambios previstos. Así se reduce la posibilidad de efectos colaterales.
- MOMENTO DE IMPLEMENTACIÓN (BINDING TIME)
  En una arquitectura que propicia modificaciones tardías en el ciclo de vida, en promedio, un cambio costará menos que en una arquitectura que fuerza que la misma modificación sea hecha más temprano.

# MODIFICABILIDAD – TÁCTICAS



# PERFORMANCE

## DEFINICIÓN

Se ocupa de  analizar cuanto tiempo le toma a un sistema responder cuando ocurre un evento.

## ÁREAS DE INTERÉS

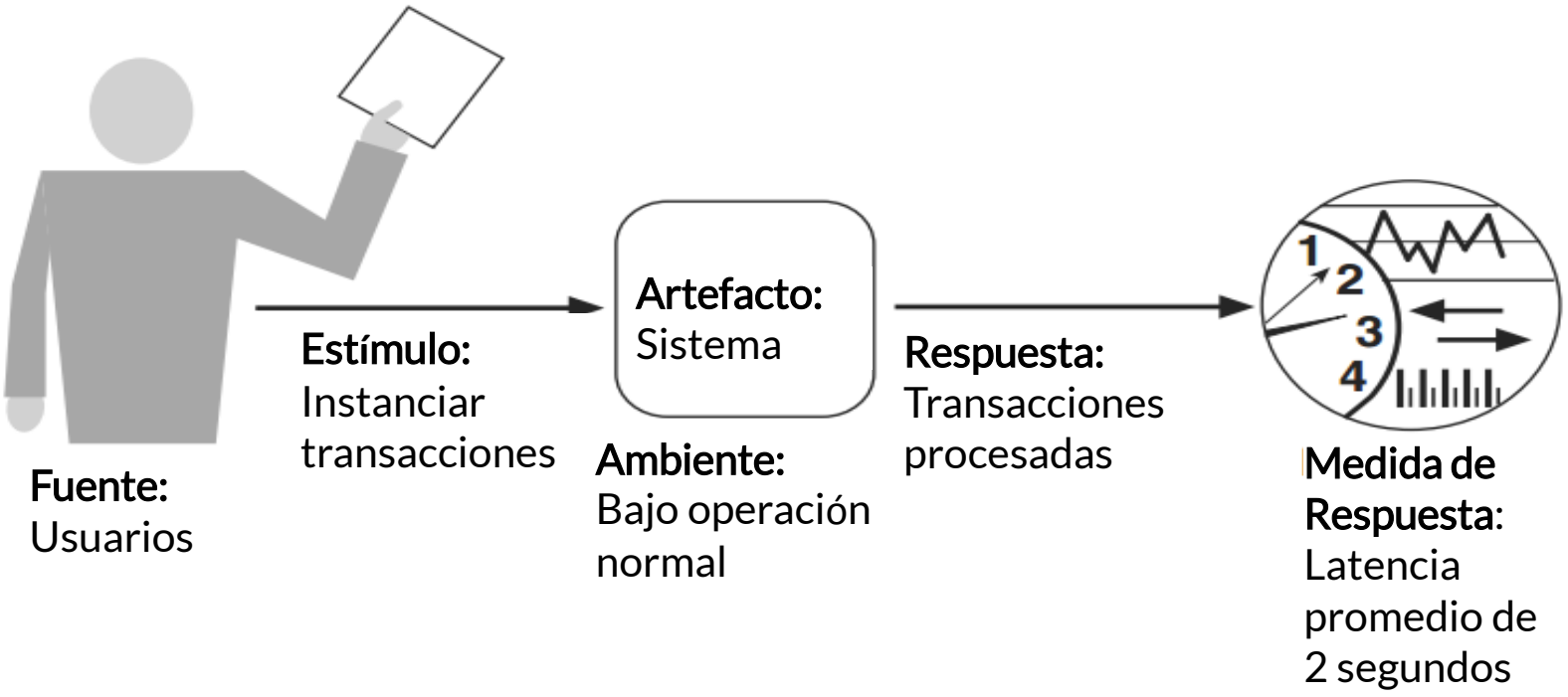
| | | |
|---|---|---|
| ¿Cuál es el origen de los eventos? | ○ usuarios<br>○ otros sistemas<br>○ el mismo sistema | |
| ¿Cómo arriban los eventos? (¿que patrón sigue el arribo?) | Periódico – cada cierto intervalo de tiempo<br>Estocástico – de acuerdo a una distribución probabilística<br>Esporádico – sin una regularidad | |
| ¿Qué variables se deben evaluar? | ○ Latencia<br>○ Momento de procesamiento<br>○ Varianza de la latencia<br>○ Datos perdidos por exceso de carga | ○ Throughput<br>○ Cantidad de eventos no procesados por exceso de carga |

# PERFORMANCE – EJEMPLO DE ESCENARIO GENERAL

| PARTE | VALORES POSIBLES |
|---|---|
| FUENTE | Una de un número independiente de fuentes posibles |
| ESTÍMULO | Eventos periódicos arriban<br>Eventos estocásticos arriban<br>Eventos esporádicos arriban |
| ARTEFACTO | El sistema, uno de sus componentes, un conjunto de componentes |
| AMBIENTE | Modo normal<br>Modo sobrecargado<br>Modo emergencia |
| RESPUESTA | ▪ Procesar el estímulo<br>▪ Cambiar el nivel de servicio |
| MEDIDA DE LA RESPUESTA | ▪ Latencia<br>▪ Deadline<br>▪ Throughput<br>▪ Varianza de latencia<br>▪ Tasa de pérdida de requerimientos<br>▪ Tasa de pérdida de datos |

# PERFORMANCE – EJEMPLO DE ESCENARIO CONCRETO

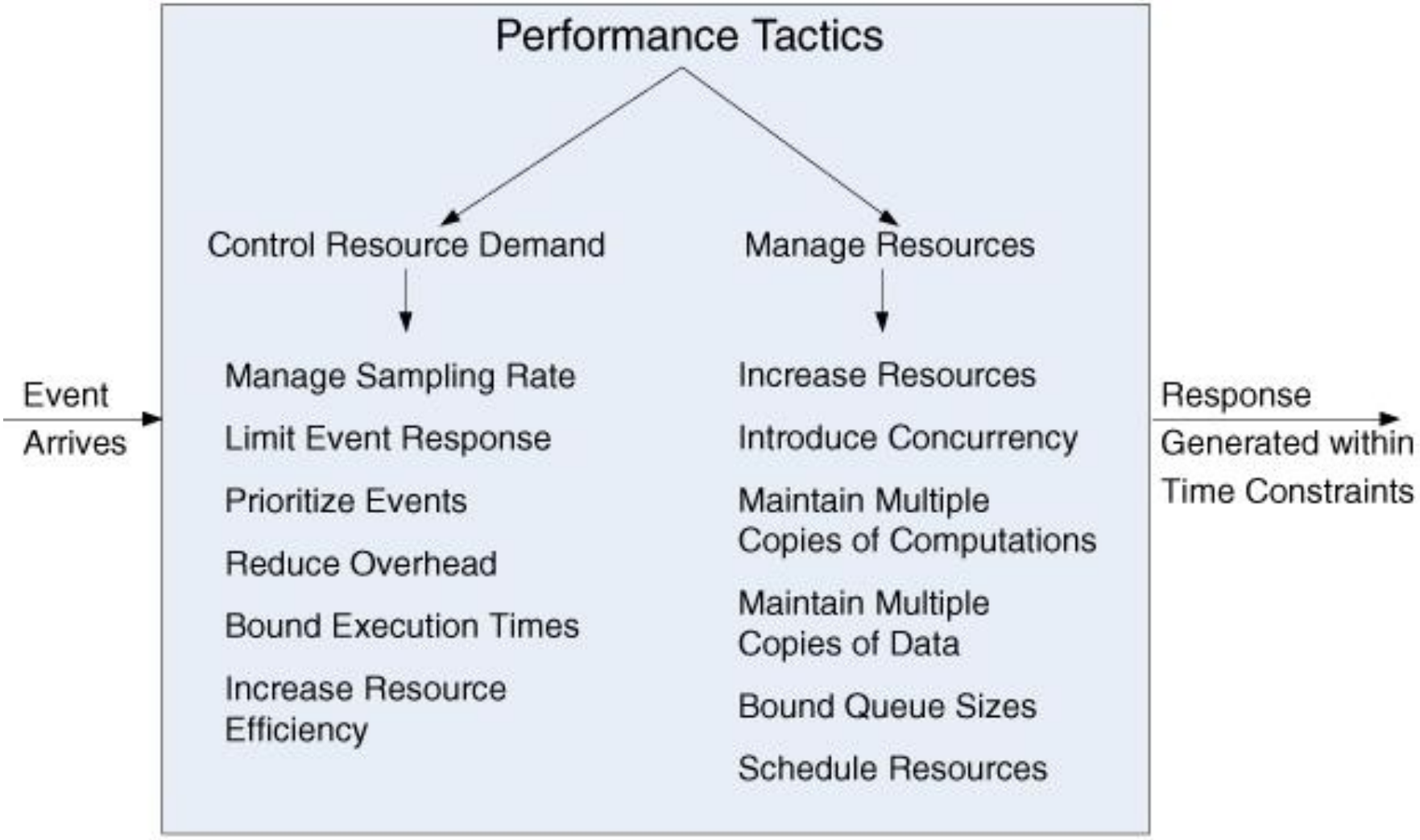
"Bajo una operación normal del sistema, las transacciones ejecutadas por los usuarios deberían resolverse, en promedio, en 2 segundos."



# PERFORMANCE – CONSIDERACIONES PARA TÁCTICAS

- TIEMPO DE PROCESAMIENTO
  Los eventos son manejados por la ejecución de uno o más componentes, cuyo tiempo utilizado es un recurso.
- CONTENCIÓN DE RECURSOS
  Muchos recursos pueden ser usados por un único cliente a la vez. Otros clientes deben esperar para acceder a ese recurso.
- TIEMPO DE BLOQUEO
  Un cómputo puede estar bloqueado debido a contención sobre otro recurso, ya que el recurso no está disponible, o a que el cómputo depende del resultado de otro cómputo que aun no está disponible.
- DISPONIBILIDAD DE RECURSOS
  Un cómputo podría estar detenido debido a la no disponibilidad de un recurso: porque el recursos está offline, o porque está atravesando una falla.
- DEPENDENCIA SOBRE OTROS MÓDULOS
  Un cómputo puede tener que esperar porque debe sincronizarse con el resultado de otro cómputo.

# PERFORMANCE – TÁCTICAS



DEMANDA DE RECURSOS CONTROLADA

## ADMINISTRAR LA FRECUENCIA DE MUESTREO

Si es posible reducir la frecuencia de muestreo a la que se captura una corriente de datos de entorno, entonces la demanda puede reducirse, normalmente con alguna pérdida de fidelidad.

Esto es común en sistemas de procesamiento de señales donde, por ejemplo, se pueden elegir diferentes códecs con diferentes velocidades de muestreo y formatos de datos. Esta elección de diseño se hace para mantener niveles predecibles de latencia.

DEMANDA DE RECURSOS CONTROLADA

## LIMITAR LA FRECUENCIA A EVENTOS

Cuando eventos discretos llegan al sistema (o elemento) demasiado rápido para ser procesados, entonces los eventos deben ser encolados hasta que puedan ser procesados.

Si se adopta esta táctica y es inaceptable perder cualquier evento, entonces debe asegurarse de que sus colas son lo suficientemente grandes para manejar el peor caso.

Si, por el contrario, elige cancelar eventos, entonces debe elegir una política para manejar esta situación: ¿Registra los eventos eliminados o simplemente los ignora? ¿Notificas a otros sistemas, usuarios o administradores?

# PERFORMANCE – EXPLICACIÓN DE TÁCTICAS 3

DEMANDA DE RECURSOS CONTROLADA

## PRIORIZAR EVENTOS

Si no todos los eventos son igualmente importantes, se puede imponer un esquema de prioridad que clasifica los eventos de acuerdo a lo importante que es para su servicio

## REDUCIR LA SOBRECARGA

El uso de intermediarios (tan importante para la modificabilidad) aumenta los recursos consumidos en el procesamiento de un flujo de eventos, y por lo tanto su eliminación mejora la latencia.

Una estrategia para reducir la sobrecarga computacional es la co-localización de recursos. La co-localización puede significar alojar componentes que cooperen en el mismo procesador para evitar el retardo de tiempo de la comunicación de la red.

# PERFORMANCE – EXPLICACIÓN DE TÁCTICAS 4

DEMANDA DE RECURSOS CONTROLADA

## TIEMPO DE EJECUCIÓN ACOTADOS

Colocar un límite en el tiempo de ejecución que se utiliza para responder a un evento.

Para algoritmos iterativos dependientes de los datos, limitar el número de iteraciones es un método para limitar los tiempos de ejecución.

## AUMENTAR LA EFICIENCIA DE LOS RECURSOS

Mejorar los algoritmos utilizados en áreas críticas disminuirá la latencia.

# PERFORMANCE – EXPLICACIÓN DE TÁCTICAS 5

ADMINISTRACIÓN DE RECURSOS

## AUMENTAR LOS RECURSOS

Por ejemplo,  procesadores más rápidos, procesadores adicionales, memoria adicional y redes más rápidas tienen el potencial de reducir la latencia.

## INTRODUCIR CONCURRENCIA

Si las solicitudes se pueden procesar en paralelo, se puede reducir el tiempo bloqueado.

## MANTENER MÚLTIPLES COPIAS DE CÁLCULO

Por ejemplo, múltiples servidores en un patrón cliente-servidor son réplicas de cálculo.

El propósito de las réplicas es reducir el tiempo que se incurriría si todos los cálculos tuvieran lugar en un único servidor.

Un equilibrador de carga es una pieza de software que asigna nuevo trabajo a uno de los servidores duplicados disponibles.

# PERFORMANCE – EXPLICACIÓN DE TÁCTICAS 6

ADMINISTRACIÓN DE RECURSOS

## MANTENER MÚLTIPLES COPIAS DE DATOS

El almacenamiento en caché es una táctica que implica mantener copias de datos (posiblemente uno es un subconjunto del otro) en almacenamiento con diferentes velocidades de acceso.

La replicación de datos implica mantener copias separadas de los datos para reducir múltiples accesos simultáneos.

## TAMAÑOS DE COLAS ACOTADOS

Esto controla el número máximo de llegadas en cola y, en consecuencia, los recursos utilizados para procesar las llegadas.

Si se adopta esta táctica, se debe adoptar una política para lo que sucede cuando el desbordamiento de colas y decidir si no responder a eventos perdidos es aceptable. Esta táctica es frecuentemente emparejada con la táctica de respuesta de evento limitada.

ADMINISTRACIÓN DE RECURSOS

## PLANIFICACIÓN DE RECURSOS

Cada vez que hay disputas para un recurso, el recurso debe planificarse. Los procesadores, búferes, y redes están planificados.

Su objetivo es entender las características del uso de cada recurso y elegir la estrategia de planificación que sea compatible con él. (FIFO, Round Robin, Importancia semántica, Importancia por deadline).

# SEGURIDAD

## DEFINICIÓN

Es una medida de la capacidad que tiene un sistema para resistir el uso no autorizado mientras continua proveyendo servicios a usuarios legítimos.

## ÁREAS DE INTERÉS

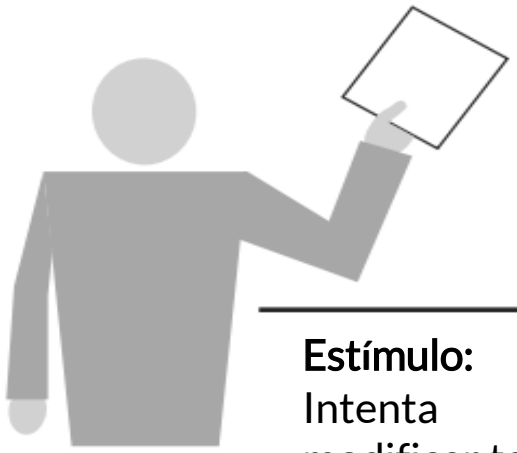
| | | |
|---|---|---|
| ¿Qué tipos de ataques pueden ocurrir? | ○ acceder a datos/servicios a los que el usuario no está autorizado<br>○ limitar el servicio a usuarios legítimos | |
| ¿Cómo se puede caracterizar la seguridad? | ○ Confidencialidad<br>○ Integridad<br>○ Disponibilidad | ○ Autenticación<br>○ No repudiación<br>○ Autorización |

# SEGURIDAD – EJEMPLO DE ESCENARIO GENERAL

| PARTE | VALORES POSIBLES |
|---|---|
| FUENTE | Individuo o sistema que está...correctamente identificado, incorrectamente identificado, o es de identidad desconocida<br>que es…interno/externo, autorizado/no autorizado<br>con acceso a…recursos limitados, vastos recursos |
| ESTÍMULO | Trata de…ver datos, modificar/eliminar datos, acceder a servicios del sistema, reducir la disponibilidad de los servicios del sistema |
| ARTEFACTO | servicios del sistema; datos dentro del sistema |
| AMBIENTE | online u offline; conectado o desconectado; detrás de un firewall o abierto |
| RESPUESTA | Autentica al usuario; oculta la identidad del usuario; bloquea/permite accesos a datos y/o servicios; garantiza o rechaza permisos para acceder a datos y/o servicios; registra los accesos por usuario; almacena datos en un formato no legible; reconoce un inexplicable aumento de demanda del servicio, informando a un usuario u otro sistema y restringiendo la disponibilidad del servicio. |
| MEDIDA DE LA RESPUESTA | Tiempo/esfuerzo/recursos necesarios para sortear medidas de seguridad con probabilidad de éxito; probabilidad de detectar ataques; probabilidad de identificar responsables de un ataque o del acceso/modificación de datos; porcentaje de servicio aun disponible ante un ataque “denial-of-services”. |

# PERFORMANCE – EJEMPLO DE ESCENARIO CONCRETO

"Un empleado descontento desde una ubicación remota intenta acceder a la tabla de tasas de pago durante la operación normal. El sistema mantiene un registro de auditoría y los datos correctos son restaurados dentro de 1 día."



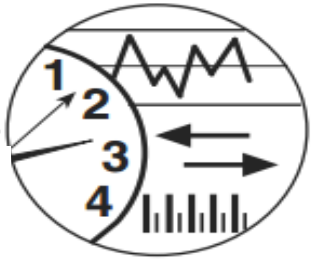
**Fuente:**
Empleado descontento conectado remotamente

**Estímulo:**
Intenta modificar tasas de pago

**Artefacto:**
Datos dentro del sistema

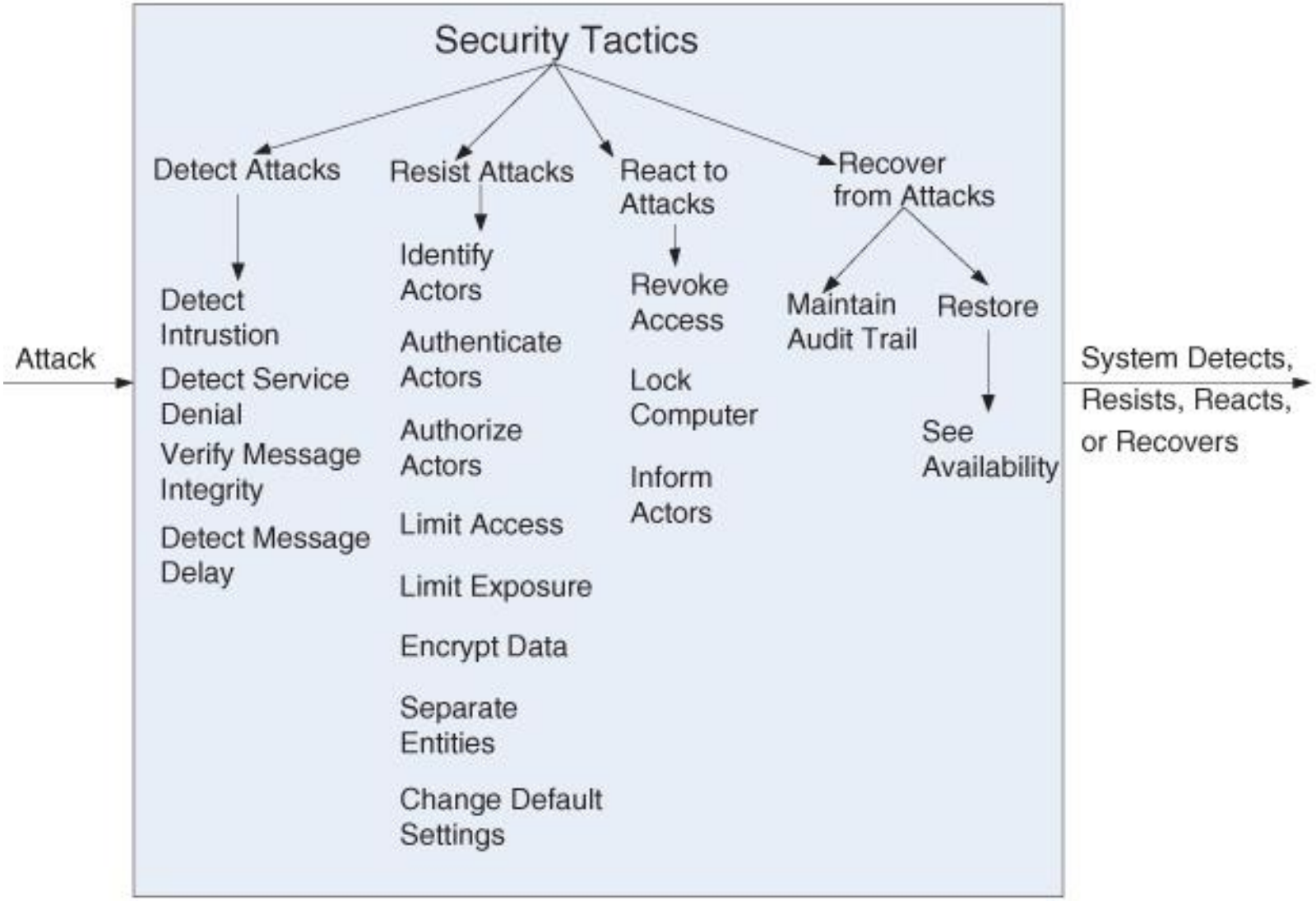
**Ambiente:**
Bajo operación normal

**Respuesta:**
El sistema mantiene registros de auditoría




**Medida de Respuesta:**
Los datos correctos son restaurados dentro de 1 día y el autor es notificado

# SEGURIDAD – TÁCTICAS



# SEGURIDAD – EXPLICACIÓN DE TÁCTICAS 1

DETECTAR ATAQUES

## DETECTAR INTRUSIÓN

Se basa en la comparación del tráfico de la red o los patrones de solicitud de servicio dentro de un sistema con un conjunto de firmas o patrones conocidos de comportamiento malicioso almacenados en una base de datos. Ejemplo: Listas negras / blancas de direcciones IP que pueden acceder.

## DETECTAR NEGACIÓN DE SERVICIO

Se basa en la comparación del patrón o la firma del tráfico de red que entra en un sistema con los perfiles históricos de ataques conocidos de denegación de servicio.

## VERIFICAR INTEGRIDAD DE MENSAJES

Esta táctica emplea técnicas tales como sumas de comprobación o valores de hash para verificar la integridad de los mensajes, archivos de recursos, archivos de implementación y archivos de configuración.

# SEGURIDAD – EXPLICACIÓN DE TÁCTICAS 2

DETECTAR ATAQUES

## DETECTAR RETRASO DE MENSAJES

Está destinado a detectar posibles ataques de intermediarios, en los que una parte maliciosa está interceptando (y posiblemente modificando) los mensajes.

Al comprobar el tiempo que se tarda en entregar un mensaje, es posible detectar el comportamiento sospechoso de tiempo, donde el tiempo que tarda en entregar un mensaje es muy variable.

RESISTIR ATAQUES

## IDENTIFICAR ACTORES

La identificación de "actores" consiste realmente en identificar la fuente de cualquier aporte externo al sistema. Los usuarios normalmente se identifican a través de ID de usuario. Otros sistemas pueden ser "identificados" a través de códigos de acceso, direcciones IP, protocolos, puertos, etc.

# SEGURIDAD – EXPLICACIÓN DE TÁCTICAS 3

RESISTIR ATAQUES

## AUTENTICAR ACTORES

La autenticación significa asegurarse que un actor (un usuario o una computadora remota) es realmente quien es o lo que pretende ser. Las contraseñas únicas, los certificados digitales y la identificación biométrica proporcionan un medio para la autenticación.

## AUTORIZAR ACTORES

La autorización significa garantizar que un actor autenticado tiene los derechos de acceso y de modificación de datos o servicios.

## ACCESO LIMITADO

Limitar el acceso a los recursos informáticos implica limitar el acceso a recursos como la memoria, conexiones de red o puntos de acceso. Por ejemplo, una zona desmilitarizada (DMZ) se utiliza cuando una organización quiere permitir que los usuarios externos accedan a ciertos servicios y no accedan a otros servicios. Se encuentra entre Internet y un cortafuegos frente a la intranet interna.

# SEGURIDAD – EXPLICACIÓN DE TÁCTICAS 4

RESISTIR ATAQUES

## LIMITAR EXPOSICIÓN

La táctica minimiza la “superficie” de ataque de un sistema. Significa tener el menor número posible de puntos de acceso para recursos, datos o servicios y reducir el número de conectores que pueden proporcionar exposición no anticipada.

## ENCRIPTAR DATOS

Los datos deben estar protegidos contra el acceso no autorizado. La confidencialidad generalmente se logra mediante la aplicación de algún tipo de cifrado a los datos ya la comunicación.

## ENTIDADES SEPARADAS

Los datos sensibles se separan frecuentemente de los datos no sensibles para reducir las posibilidades de ataque de aquellos que tienen acceso a datos no sensibles.

# SEGURIDAD – EXPLICACIÓN DE TÁCTICAS 5

RESISTIR ATAQUES

## CAMBIAR SETEOS POR DEFECTO

Muchos sistemas tienen valores predeterminados asignados cuando se entrega el sistema. Obligar al usuario a cambiar esas configuraciones evitará que los atacantes accedan al sistema a través de configuraciones que, por lo general, están disponibles públicamente. Ejemplo, router WIFI

REACCIONAR A ATAQUES

## REVOCAR EL ACCESO

Si el sistema o el administrador del sistema cree que un ataque está en curso, el acceso puede estar severamente limitado a recursos sensibles, incluso para usuarios y usos normalmente legítimos.

# SEGURIDAD – EXPLICACIÓN DE TÁCTICAS 6

REACCIONAR A ATAQUES

## BLOQUEAR COMPUTADOR

Los intentos fallidos de inicio de sesión fallidos pueden indicar un posible ataque. Muchos sistemas limitan el acceso desde una computadora en particular si hay repetidos intentos fallidos de acceder a una cuenta desde esa computadora. Los usuarios legítimos pueden cometer errores al intentar iniciar sesión. Por lo tanto, el acceso limitado sólo puede ser durante un cierto período de tiempo.

## INFORMAR ACTORES

Los ataques en curso pueden requerir la acción de operadores, otro personal o sistemas cooperantes. Tal personal o sistemas -el conjunto de actores relevantes- debe ser notificado cuando el sistema ha detectado un ataque.

RECUPERACIÓN DE UN ATAQUE

## MANTENER RASTROS DE AUDITORÍA

Se audita, es decir se mantiene un registro de las acciones del usuario y del sistema y sus efectos para ayudar a rastrear las acciones de un atacante y para identificarlo.

## RECUPERACIÓN

Parte de la recuperación es la restauración de los servicios. Por ejemplo, servidores adicionales o conexiones de red pueden mantenerse en reserva para tal propósito.

Puesto que un ataque exitoso puede ser considerado como una especie de fracaso, el conjunto de tácticas de disponibilidad que se ocupan de la recuperación de un fracaso puede ser llevado a cabo para este aspecto de la seguridad también.

# CONTENIDO

# TESTEABILIDAD

## DEFINICIÓN

Es el grado en el cual un artefacto de software (sistema, módulo, clase, etc.) soporta testing en un contexto de test dado. Si la testeabilidad del artefacto de software es alta, entonces encontrar defectos a través del testing es más sencillo.

## ÁREAS DE INTERÉS

| | |
|---|---|
| ¿Qué tipos de tests se deben hacer? | Se busca principalmente, la posibilidad de realizar test automatizados: tests unitarios, de integración de interfaces de usuarios |
| ¿Qué significa que un artefacto de software sea testeable? | Que se pueda:<br>○ controlar las entradas del componente, probando su estado interno<br>○ observar las salidas |

# TESTEABILIDAD – EJEMPLO DE ESCENARIO GENERAL

| PARTE | VALORES POSIBLES |
|---|---|
| FUENTE | Testers unitarios; testers de integración, tester de sistema, tester de aceptación, usuario finales del sistema (ya sea de manera manual o automática) |
| ESTÍMULO | Un conjunto de tests es ejecutado debido a:<br>○ Incremento de código terminado o integración de subsistema terminada<br>○ Sistema entregado |
| ARTEFACTO | La porción del sistema siendo testeada |
| AMBIENTE | ○ En tiempo de desarrollo<br>○ En tiempo de compilación<br>○ En tiempo de deployment<br>○ En tiempo de ejecución |
| RESPUESTA | ○ Ejecutar un conjunto de tests y capturar el resultado<br>○ Capturar las actividades que resultaron en una falla<br>○ Controlar y monitorear el estado del sistema |
| MEDIDA DE LA RESPUESTA | ○ Porcentaje de cobertura del código ejecutable<br>○ Probabilidad de falla, si la falla existe<br>○ Tiempo para realizar los test<br>○ Longitud de la cadena de dependencias más larga en un test<br>○ Cantidad de tiempo para preparar un ambiente de test |

# TESTEABILIDAD – EJEMPLO DE ESCENARIO CONCRETO

"Un tester unitario (desarrollador) terminó una unidad de código durante el desarrollo y realiza una secuencia de tests cuyos resultados son capturados logrando una cobertura del 85% en menos de 3 horas."

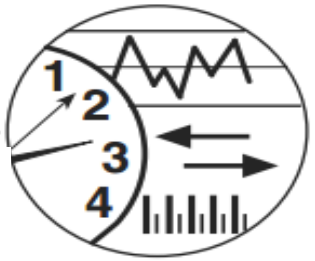
**Fuente:**
Tester unitario

**Estímulo:**
Unidad de código terminada

**Artefacto:**
Unidad de código

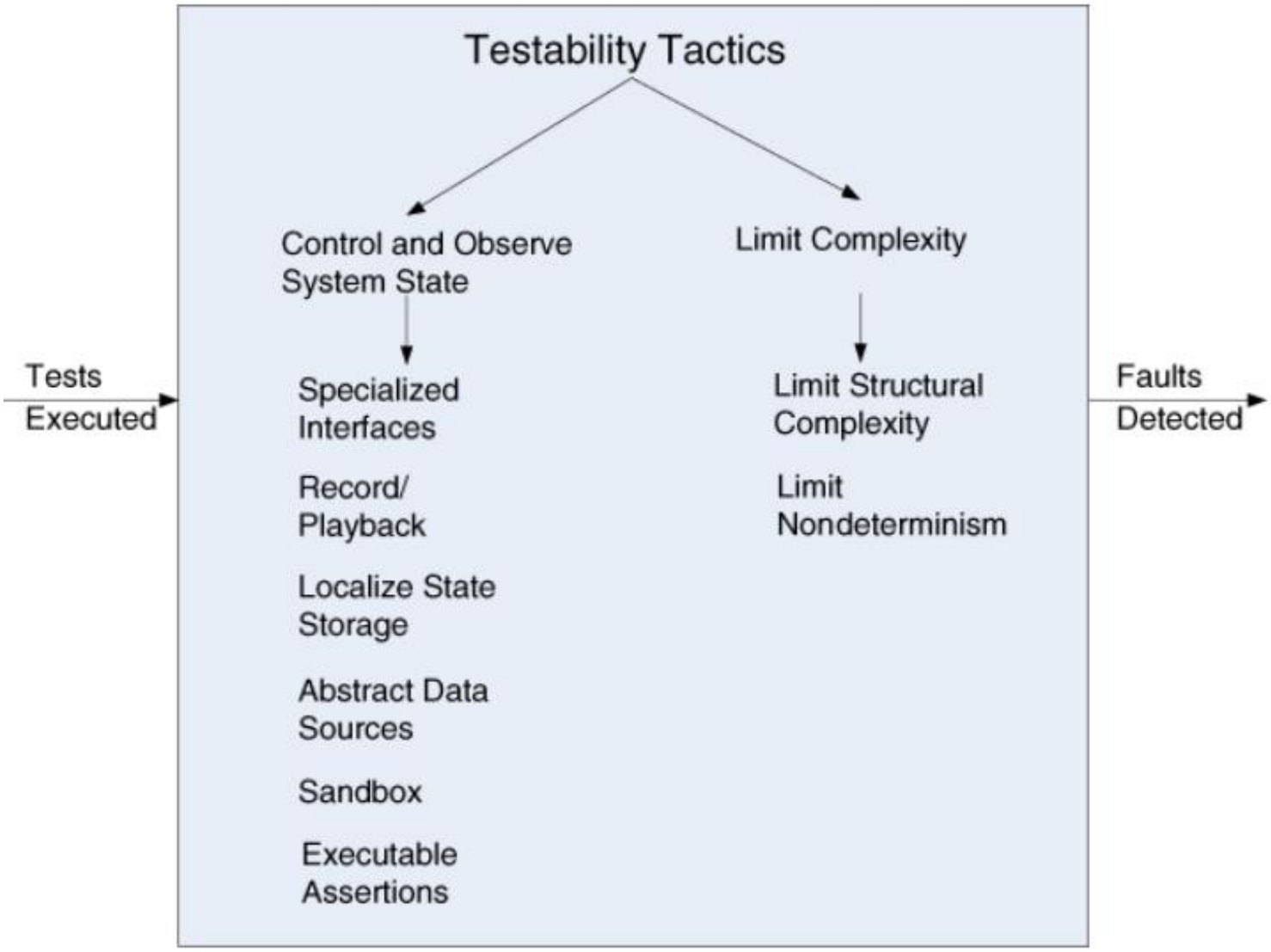
**Ambiente:**
Desarrollo

**Respuesta:**
Resultados capturados




**Medida de Respuesta:**
La cobertura del código del 85% es realizada en 3 horas

# TESTEABILIDAD – TÁCTICAS



# TESTEABILIDAD – EXPLICACIÓN DE TÁCTICAS 1

CONTROLAR Y OBSERVAR EL ESTADO DEL SISTEMA

Estas tácticas hacen que un componente mantenga algún tipo de información de estado, permita que los testeadores asignen un valor a esa información de estado y / o que esa información sea accesible a los testeadores bajo demanda.

La información de estado puede ser un estado operativo, el valor de alguna variable clave, carga de rendimiento, pasos intermedios del proceso o cualquier otra cosa útil para volver a crear el comportamiento del componente.

# TESTEABILIDAD – EXPLICACIÓN DE TÁCTICAS 2

CONTROLAR Y OBSERVAR EL ESTADO DEL SISTEMA

## INTERFACES ESPECIALIZADAS

Tener interfaces de prueba especializadas permite controlar o capturar valores de variables para un componente, ya sea a través de un modo de prueba o mediante una ejecución normal:

- un método set y get para variables, modos o atributos importantes
- un método de informe que devuelva el estado completo del objeto
- un método de reinicio para establecer el estado interno en un estado interno especificado
- un método para activar la salida detallada, varios niveles de registro de eventos, instrumentación de rendimiento o supervisión de recursos.

## GRABAR Y REPRODUCIR

El estado que causó una falla es a menudo difícil de volver a crear.
Grabar el estado cuando “se cruza” una interfaz permite que ese estado se utilice para "reproducir el sistema de nuevo" y para volver a crear el fallo.

# TESTEABILIDAD – EXPLICACIÓN DE TÁCTICAS 3

CONTROLAR Y OBSERVAR EL ESTADO DEL SISTEMA

## LOCALIZAR EL ESTADO DE ALMACENAMIENTO

Para iniciar un sistema, un subsistema o un módulo en un estado arbitrario para una prueba, es más conveniente si ese estado se almacena en un solo lugar. Por el contrario, si el estado es enterrado o distribuido, esto se vuelve difícil, si no imposible.

Una máquina de estados podría ser muy útil para estos casos

## FUENTES DE DATOS ABSTRACTAS

Al igual que controlar el estado de un programa, poder controlar fácilmente los datos de entrada facilita la prueba. Hacer abstracción de las interfaces, permite sustituir los datos de prueba más fácilmente. Por ejemplo, poder reemplazar fácilmente la base de datos real para que direccione a una base de datos de prueba.

# TESTEABILIDAD – EXPLICACIÓN DE TÁCTICAS 4

CONTROLAR Y OBSERVAR EL ESTADO DEL SISTEMA

## SANDBOX

"Sandboxing" se refiere a aislar una instancia del sistema del mundo real para permitir la experimentación que no está restringida por la preocupación de tener que deshacer las consecuencias del experimento.
Las pruebas son ayudadas por la capacidad de operar el sistema de tal manera que no tiene consecuencias permanentes, o que cualquier consecuencia pueda revertirse.

El framework Spring, popular para Java, trae un conjunto de utilidades de prueba que soportan esto. Las pruebas se ejecutan como una "transacción", que se deshace al final.

## ASERCIONES EJECUTABLES

Las aserciones son (generalmente) codificadas manualmente y colocadas en los lugares deseados para indicar cuándo y dónde un programa está en un estado defectuoso. Las aserciones a menudo están diseñadas para comprobar que los valores de datos satisfacen restricciones especificadas. Esto se traduce en una observabilidad creciente, cuando una aserción se señala como fracasada.

# TESTEABILIDAD – EXPLICACIÓN DE TÁCTICAS 5

LIMITAR LA COMPLEJIDAD

## LIMITAR COMPLEJIDAD ESTRUCTURAL

Esta táctica trata de evitar o resolver problemas cíclicos de dependencias entre componentes, aislando y encapsulando las dependencias en el entorno externo, y, en general, reduciendo las dependencias entre componentes. Por ejemplo, reduciendo el número de accesos externos a los datos públicos de un módulo.
Tener alta cohesión, bajo acoplamiento y separación de intereses -todas tácticas de modificabilidad- también puede ayudar con la testeabilidad.

## LIMITAR EL NO-DETERMINISMO

Involucra encontrar todas las fuentes de no determinismo; como por ejemplo, el paralelismo sin restricciones, y eliminarlos en la medida de lo posible.

El no determinismo es una forma muy perniciosa de comportamiento complejo.

# CASO DE ESTUDIO – EL EJÉRCITO DE SIMIOS DE NETFLIX

- Netfix está montado sobre la nube de Amazon (AWS), más precisamente sobre instancias de EC2 (servidores virtuales).
- Netflix pondera la Testeabilidad de su sistema para asegurar la calidad de su servicio
- Utiliza un “ejército de simios” como para parte de su proceso de testing.
  - ✓ Chaos Monkey: Aleatoriamente mata procesos en el sistema ejecutando para monitorear el efecto de un proceso fallido.
  - ✓ Latency Monkey: Introduce retardos artificiales en la capa de comunicación “cliente-servidor” para simular degradación.
  - ✓ Conformity Monkey: Busca instancias que no adhieren a las buenas prácticas y las baja (por ej., instancias que no pertenecen a un auto-scaling group).
  - ✓ Doctor Monkey: Monitorea la salud de las instancias de EC2.
  - ✓ Janitor Monkey: Asegura que en ambiente de Netflix en la nube no desperdicia recursos. Donde encuentra un recurso no usado, lo libera.
  - ✓ Security Monkey: Busca violaciones o vulnerabilidades de seguridad. Si las encuentra, las elimina. También chequea que los certificados SSL no se venzan.
  - ✓ 10-18 Monkey: Detecta problemas de configuración de localización e i18n.

# USABILIDAD

## DEFINICIÓN

Es la facilidad con la cual las personas pueden utilizar un sistema con el fin de alcanzar un objetivo concreto

## ÁREAS DE INTERÉS

Qué puede hacer el sistema para que a un usuario que no está familiarizado con él pueda aprender a usarlo más fácilmente?
APRENDIZAJE DE CARACTERÍSTICAS DEL SISTEMA

Qué puede hacer el sistema para mejorar la eficiencia en la operatoria de un usuario?
USO EFICIENTE DEL SISTEMA

Qué puede hacer el sistema para minimizar el impacto de un error de usuario?
MINIMIZAR EL IMPACTO DE ERRORES

Cómo puede el usuario (o el sistema en si mismo) adaptarse para hacer la tarea más fácil?
ADAPTACION DEL SISTEMA A LAS NECESIDADES DEL USUARIO

Qué hace el sistema para darle confianza al usuario que está ejecutando la acción correcta?
INCREMENTAR LA CONFIANZA Y SATISFACCIÓN

# USABILIDAD – EJEMPLO DE ESCENARIO GENERAL 1

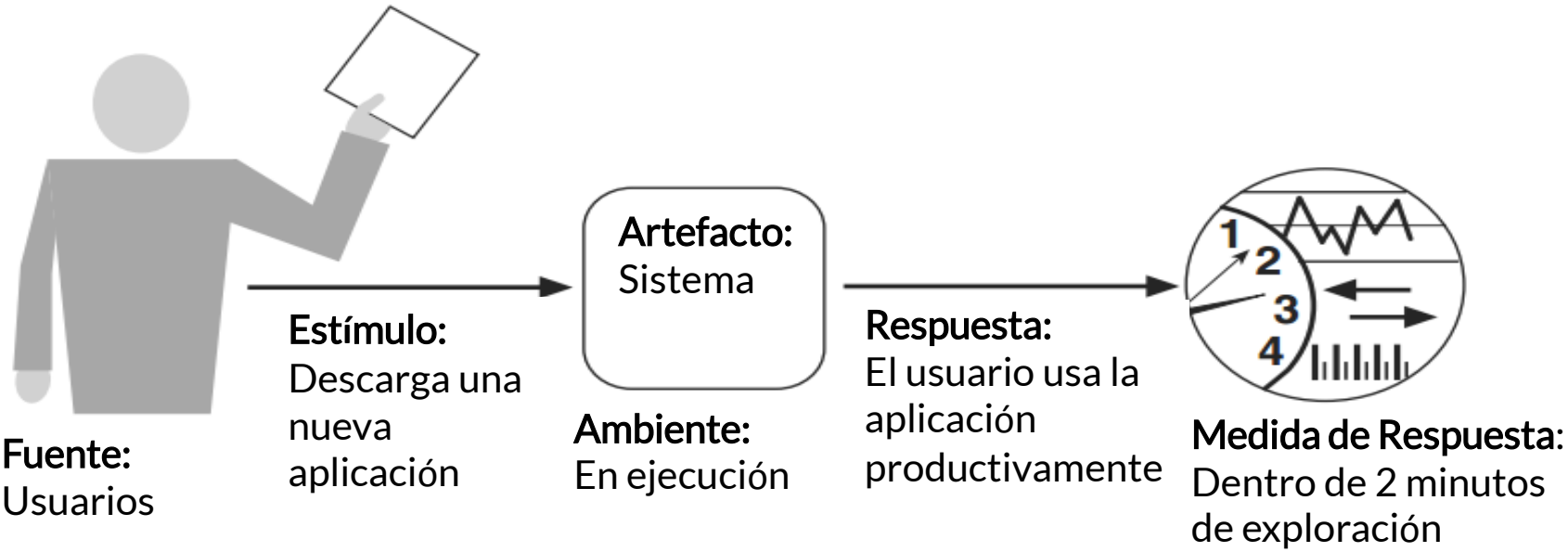
| PARTE | VALORES POSIBLES |
|---|---|
| FUENTE | Usuario final |
| ESTÍMULO | Quiere… aprender las características del sistema; usar el sistema eficientemente; minimizar el impacto de errores de usuario; adaptar el sistema a sus necesidades; configurar el sistema |
| ARTEFACTO | Sistema |
| AMBIENTE | ○ En tiempo de ejecución ○ En tiempo de configuración |

# USABILIDAD – EJEMPLO DE ESCENARIO GENERAL 2

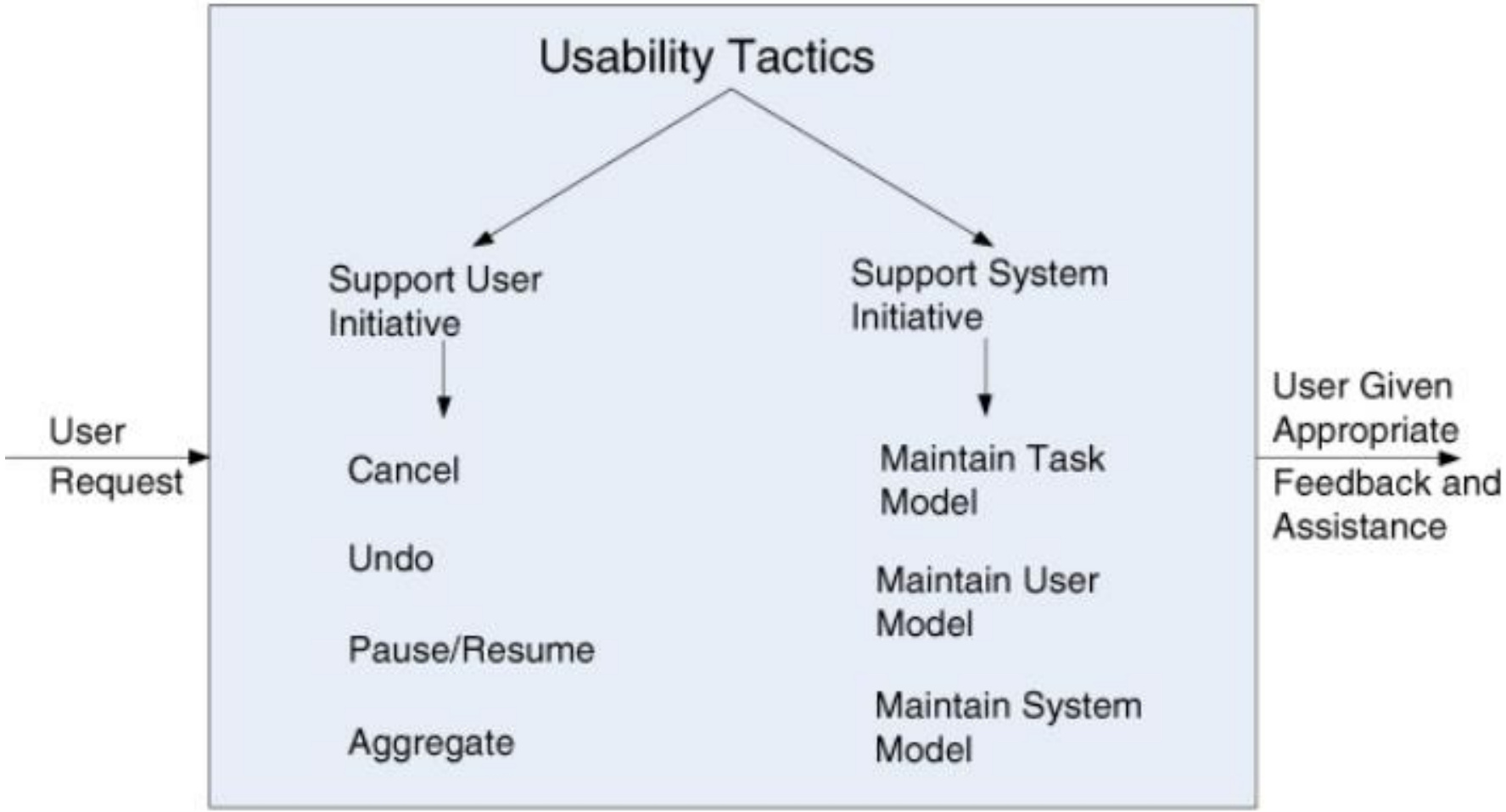
| PARTE | VALORES POSIBLES |
|---|---|
| RESPUESTA | ○ Para soportar el aprendizaje de las características del sistema<br>Sistema de ayuda sensible al contexto; UI familiar para el usuario<br>○ Para soportar el uso del sistema eficientemente<br>Agregación de datos/comandos; reuso de datos y comandos ya ingresados; soporte para una navegación eficiente; distintas vistas con operaciones consistentes;<br>○ Para minimizar el impacto de errores de usuario<br>Deshacer, rehacer y recuperarse de una falla del sistema; reconocer y corregir un error de usuario; recuperar password olvidada<br>○ Para adaptar el sistema<br>Personalización; Internacionalización<br>○ Para sentirse cómodo<br>Mostrar el estado del sistema; trabajar al ritmo del usuario |
| MEDIDA DE LA RESPUESTA | Tiempo de la tarea; cantidad de errores; cantidad de problemas resueltos; satisfacción del usuario; tasa de operaciones exitosas vs operaciones no exitosas; interacción necesaria para completar una tarea |

# USABILIDAD – EJEMPLO DE ESCENARIO CONCRETO

"Un usuario descarga una nueva aplicación y está usándola productivamente luego de 2 minutos de experimentación."



# USABILIDAD – TÁCTICAS



# USABILIDAD – EXPLICACIÓN DE TÁCTICAS 1

## APOYAR INICIATIVAS DEL USUARIO

### CANCELAR

Cuando el usuario emite una orden de cancelación, el sistema debe estar “escuchando” (por lo tanto, existe la responsabilidad de tener un oyente constante que no está bloqueado por las acciones de lo que se está cancelando).

- el comando que se cancela debe ser terminado
- cualquier recurso que esté siendo utilizado por el comando cancelado debe ser liberado
- los componentes que colaboran con el comando cancelado deben estar informados para que también puedan tomar las medidas apropiadas.

# USABILIDAD – EXPLICACIÓN DE TÁCTICAS 2

## APOYAR INICIATIVAS DEL USUARIO

### DESHACER

Para soportar la capacidad de deshacer, el sistema debe mantener una cantidad suficiente de información sobre el estado del sistema para poder restaurar un estado anterior, a petición del usuario. Tal registro puede ser en forma de instantáneas de estado - por ejemplo, puntos de control; o como un conjunto de operaciones reversibles.

No todas las operaciones pueden invertirse fácilmente: por ejemplo, cambiar todas las ocurrencias de la letra "a" a la letra "b" en un documento no se puede revertir cambiando todas las instancias de "b" a "a", porque algunas de esas instancias de "b" puede haber existido antes del cambio original. En tal caso, el sistema debe mantener un registro más elaborado del cambio.

Por supuesto, algunas operaciones, como sonar una campana, no se pueden deshacer.

# USABILIDAD – EXPLICACIÓN DE TÁCTICAS 3

APOYAR INICIATIVAS DEL USUARIO

## PAUSAR/REANUDAR

Cuando un usuario ha iniciado una operación de larga duración -por ejemplo, descargando un archivo grande o un conjunto de archivos de un servidor- a menudo es útil proporcionar la capacidad de pausar y reanudar la operación.

La detención efectiva de una operación de larga duración requiere la capacidad de liberar temporalmente recursos para que puedan ser reasignados a otras tareas.

## AGREGADO

Cuando un usuario está realizando operaciones repetitivas, o operaciones que afectan a un gran número de objetos de la misma manera, es útil proporcionar la capacidad de agrupar los objetos de nivel inferior en un solo grupo, de modo que la operación se pueda aplicar al grupo. Esto libera al usuario de la carga (y el potencial de errores) de hacer la misma operación repetidamente.
Por ejemplo, agregar todos los objetos de las transparencias y cambiar la fuente del texto a 14 puntos.

# USABILIDAD – EXPLICACIÓN DE TÁCTICAS 4

APOYAR INICIATIVAS DEL SISTEMA

## MANTENER EL MODELO DE TAREAS

El modelo de tarea se utiliza para determinar el contexto, para que el sistema pueda tener alguna idea de lo que el usuario está intentando hacer y de este modo proporcionar asistencia.
Por ejemplo, saber que las oraciones comienzan con letras mayúsculas permitiría que una aplicación corrija una letra minúscula en esa posición.

## MANTENER EL MODELO DEL USUARIO

Este modelo representa explícitamente el conocimiento del usuario del sistema, el comportamiento del usuario en términos de tiempo de respuesta esperado y otros aspectos específicos de un usuario o una clase de usuarios. Un modelo también puede controlar la cantidad de asistencia y sugerencias proporcionadas automáticamente a un usuario.

Un caso especial de esta táctica se encuentra comúnmente en la personalización de la interfaz de usuario, en la que un usuario puede modificar explícitamente el modelo de usuario del sistema. Ejemplo: Las sugerencias que hacen los anuncios Netflix o Goggle.

APOYAR INICIATIVAS DEL SISTEMA

## MANTENER EL MODELO DEL SISTEMA

Aquí el sistema mantiene un modelo explícito de sí mismo. Esto se utiliza para determinar el comportamiento esperado del sistema, de modo que se pueda dar al usuario la información adecuada.

Una manifestación común de un modelo de sistema es una barra de progreso que predice el tiempo necesario para completar la actividad actual.

# CONFIABILIDAD

## DEFINICIÓN

Es la capacidad de un sistema de mantenerse operativo en el tiempo.

La probabilidad que tiene el sistema de realizar su función sin fallas durante un período de tiempo.

## ÁREAS DE INTERÉS

MEDIDAS
- Tiempo medio entre fallas (MTBF)
- Porcentaje de operaciones que se ejecutan correctamente en un período

Esta ligado a otros atributos de calidad:

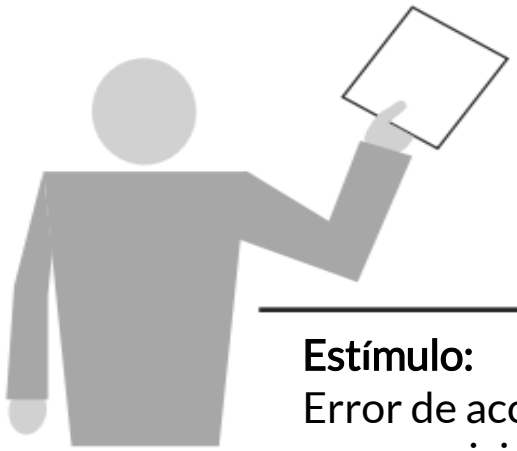
| | |
|---|---|
| ○ DISPONIBILIDAD | Es la probabilidad que tiene el Sistema de estar operativo en un determinado punto en el tiempo |
| ○ TOLERANCIA A FALLOS | La capacidad para mantener un determinado nivel de rendimiento en casos de fallas de software o de violación a su interfaz |
| ○ RECUPERABILIDAD | La capacidad de restablecer un nivel de rendimiento determinado y recuperar los datos directamente afectados en caso de falla |
| ○ MADUREZ | La capacidad de evitar fallas como resultado de defectos en el software. |

# CONFIABILIDAD – EJEMPLO DE ESCENARIO GENERAL

| PARTE | VALORES POSIBLES |
|---|---|
| FUENTE | Interna al sistema; externa al sistema |
| ESTÍMULO | Defecto: datos erróneos, falla de hardware, bug en el código, problemas de ambiente |
| ARTEFACTO | Procesadores del sistema, canales de comunicación, almacenamiento persistente, procesos, datos |
| AMBIENTE | Modo Normal, Modo Sobrecargado |
| RESPUESTA | El sistema detectará un evento y haría una o varias de las siguientes acciones:<br>▪ registrarlo<br>▪ notificar a quienes corresponda, incluyendo el usuario y otros sistemas<br>▪ tomar las acciones necesarias para dejar los datos consistentes<br>▪ reintentar la operación (probablemente con algún delay) |
| MEDIDA DE LA RESPUESTA | Tiempo medio entre fallas<br>Porcentaje de operaciones que se ejecutan correctamente en un periodo<br>Tiempo que el Sistema puede funcionar en modo sobrecargado |

# CONFIABILIDAD – EJEMPLO DE ESCENARIO CONCRETO

"En AWS (la nube de Amazon) las conexiones a los servicios de AWS pueden fallar eventual y esporádicamente. Ante una falla de la infraestructura de AWS, el sistema debe reintentar la transacción en 5, 10, 20, 40... segundos hasta que logre ejecutarse exitosamente."



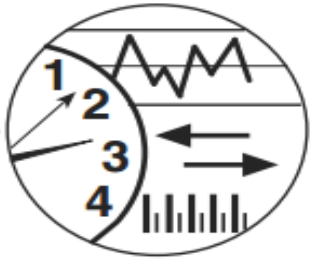
**Fuente:** Infraestructura de AWS

**Estímulo:** Error de acceso a un servicio de AWS

**Artefacto:** Proceso

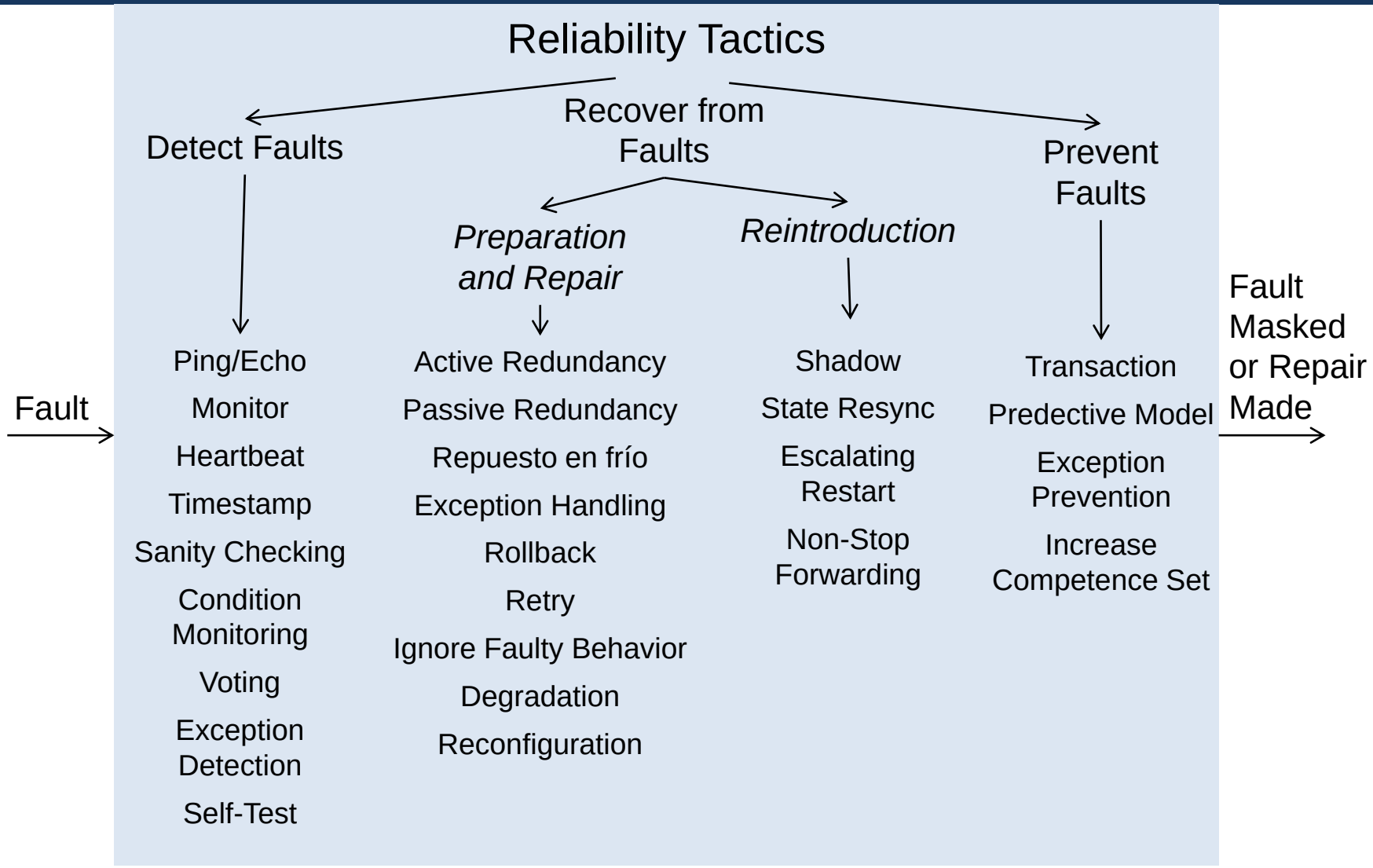
**Ambiente:** Operación normal

**Respuesta:** Reintentar la transacción hasta tener éxito. Frecuencia de reintento: 5, 10, 20, 40, … segundos




**Medida de Respuesta:** Sin transacciones fallidas

# CONFIABILIDAD – TÁCTICAS



# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 1

DETECTAR FALLAS

## PING/ECHO

Par de mensajes asincrónicos de pedido/respuesta intercambiados entre nodos, usado para determinar si se alcanza el otro nodo y la demora asociada al camino de la red.

## MONITOR

Un monitor es un componente que se utiliza para supervisar el estado de salud de varias otras partes del sistema: procesadores, procesos, E / S, memoria, etc.

## HEARTBEAT MONITOR

Es un mecanismo de detección de fallos que emplea un intercambio periódico de mensajes entre un monitor del sistema y un proceso que se supervisa.
La diferencia entre heartbeat monitor y ping/eco es quien tiene la responsabilidad de iniciar el control de salud, el monitor o el componente en sí mismo.

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 2

DETECTAR FALLAS

## ESTAMPILLA DE TIEMPO

Esta táctica se utiliza para detectar secuencias de eventos incorrectas, principalmente en sistemas distribuidos de paso de mensajes.

## CONTROL DE SANIDAD

Comprueba la validez o razonabilidad de operaciones o salidas específicas de un componente.

## MONITOREO DE CONDICIÓN

Implica verificar las condiciones en un proceso o dispositivo, o validar suposiciones hechas durante el diseño (sumas de comprobación).

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 3

DETECTAR FALLAS

## VOTACIÓN

La realización más común de esta táctica se denomina redundancia triple modular (TMR), que emplea tres componentes que hacen lo mismo, cada uno de los cuales recibe entradas idénticas, y transmite su salida a la lógica de votación, utilizada para detectar cualquier inconsistencia entre los tres estados de salida.

## DETECCIÓN DE EXCEPCIÓN

Se refiere a la detección de una condición del sistema que altera el flujo normal de ejecución.

## AUTO-TESTEO

Los componentes (o, más probablemente, subsistemas enteros) pueden ejecutar procedimientos para comprobar si funcionan correctamente.

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 4

PREPARACIÓN Y REPARACIÓN

## REDUNDANCIA ACTIVA (REPUESTO DINÁMICO)

Esto se refiere a una configuración en la que todos los nodos de un grupo de protección reciben y procesan entradas idénticas en paralelo, permitiendo que los repuestos redundantes mantengan el estado síncrono con los nodos activos.

## REDUNDANCIA PASIVA (REPUESTO CALIENTE)

Esto hace referencia a una configuración en la que sólo los miembros activos del grupo de protección procesan el tráfico de entrada. Una de sus funciones es proporcionar a los repuestos redundantes con actualizaciones de estado periódicas.

## REPUESTO (REPUESTO FRÍO)

Repuesto en frío se refiere a una configuración en la que los repuestos redundantes de un grupo de protección permanecen fuera de servicio hasta que se produce un fallo.

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 5

PREPARACIÓN Y REPARACIÓN

## MANEJO DE EXCEPCIONES

Una vez que se ha detectado una excepción, el sistema debe manejarlo de alguna manera.

## ROLLBACK

Esta táctica permite al sistema volver a un estado “bueno” conocido anterior. Una vez alcanzado el “buen” estado, la ejecución puede continuar.

## REINTENTAR

La táctica de reintento asume que la falla que causó un fallo es transitoria y reintentar la operación puede conducir al éxito. Esta táctica se utiliza en las redes y en servidores donde los fallos son esperados y comunes.

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 6

PREPARACIÓN Y REPARACIÓN

## IGNORAR EL COMPORTAMIENTO DEFECTUOSO

Esta táctica requiere ignorar los mensajes enviados desde una fuente en particular cuando se determina que esos mensajes son falsos.

## DEGRADACIÓN

Ante la presencia de fallos de componentes, se mantienen las funciones más críticas del sistema, reduciendo las funciones menos críticas.

## RECONFIGURACIÓN

La reconfiguración intenta recuperar el sistema de los fallos de los componentes mediante la reasignación de responsabilidades a los recursos (potencialmente restringidos) en funcionamiento, manteniendo al mismo tiempo la mayor funcionalidad posible.

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 6

REINTRODUCCIÓN

## SOMBRA

La táctica se refiere a la operación de un componente actualizado previamente o en servicio actualizado en un "modo de sombra" durante un período de tiempo predefinido antes de revertir el componente de nuevo a un rol activo. Durante esta duración su comportamiento puede ser monitoreado para la corrección y puede repoblar su estado de forma incremental.

## RESINCRONIZACIÓN DE ESTADO

Es una táctica de reintroducción asociada a la tácticas de redundancia activa y de redundancia pasiva de Preparación y Reparación.

## RE-ARRANQUE PROGRESIVO

Es una táctica que permite al sistema recuperarse de fallas variando la granularidad de los componentes reiniciados y minimizando el nivel de servicio afectado. Es capaz de degradar los servicios que proporciona mientras mantiene el soporte para aplicaciones de misión crítica o aplicaciones críticas.

REINTRODUCCIÓN

## NON-STOP FORWARDING

Es un concepto que se originó en el diseño del enrutador. En este diseño, la funcionalidad se divide en dos partes:

1) supervisora o de control -que gestiona la conectividad y la información de enrutamiento
2) el plano de datos -que realiza el trabajo real de enrutamiento de paquetes del emisor al receptor.

Si un enrutador experimenta el fallo de un supervisor activo, puede continuar enviando paquetes a lo largo de rutas conocidas.

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 8

PREVENIR FALLAS

## TRANSACCIÓN

Los sistemas dirigidos a servicios de alta disponibilidad aprovechan la semántica transaccional para asegurar que los mensajes asincrónicos intercambiados entre componentes distribuidos sean atómicos, consistentes, aislados y duraderos. Estas cuatro propiedades se denominan "propiedades ACID". La realización más común de la táctica de transacciones es el protocolo de "confirmación de dos fases" (2PC). Esta táctica impide condiciones causadas por 2 procesos que intentan actualizar el mismo elemento de datos.

## MODELO PREDICTIVO

Se emplea un modelo predictivo, cuando se combina con un monitor, para monitorear el estado de salud de un proceso del sistema para asegurar que el sistema está operando dentro de sus parámetros operativos nominales y para tomar medidas correctivas cuando se detectan condiciones que predicen el futuro probable de fallas.

# CONFIABILIDAD – EXPLICACIÓN DE TÁCTICAS 9

PREVENIR FALLAS

## PREVENCIÓN DE EXCEPCIONES

Esta táctica se refiere a las técnicas empleadas con el fin de evitar que se produzcan excepciones del sistema. Por ejemplo, el uso de semáforos para evitar problemas de concurrencia.

## AUMENTAR EL CONJUNTO DE COMPETENCIAS

El conjunto de competencias de un programa es el conjunto de estados en los que es "competente" para operar. Significa diseñarlo para manejar más casos-fallas-como parte de su funcionamiento normal.

**Elsa Estevez**
**ece@cs.uns.edu.ar**